ATELIERS

Tâche 1 : Surveiller le système

1. Activer les principales commandes de surveillance natives et observer les types d'informations recueillies.

```
[root@linux1 ~]# uptime
 12:46:29 up 13 days, 14:13,  2 users,  load average: 0.00, 0.00, 0.00
[root@linux1 ~]# free
             total       used       free     shared    buffers     cached
Mem:        449908     444876       5032          0      62376     309356
-/+ buffers/cache:      73144     376764
Swap:       522072          0     522072
[root@linux1 ~]# vmstat
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 0  0      0   5032  62380 309356    0    0     1     1   27    33  0  0 100  0
[root@linux1 ~]# vmstat 5 3
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 0  0      0   5032  62384 309356    0    0     1     1   27    33  0  0 100  0
 0  0      0   5032  62384 309356    0    0     0     6 1042   120  0  0 100  0
 0  0      0   5032  62384 309356    0    0     0     0 1061   167  0  0 100  0
[root@linux1 ~]# w
 12:46:56 up 13 days, 14:13,  2 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1     -                 22Jan07 12days  0.01s  0.01s -bash
root     pts/1    raichu.pokemon    Tue15    0.00s  0.27s  0.00s w
[root@linux1 ~]# ps aux | head
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1   3084   552 ?        S    Jan21   0:00 init [5]
root         2  0.0  0.0      0     0 ?        SN   Jan21   0:14 [ksoftirqd/0]
root         3  0.0  0.0      0     0 ?        S<   Jan21   0:00 [events/0]
root         4  0.0  0.0      0     0 ?        S<   Jan21   0:00 [khelper]
root         5  0.0  0.0      0     0 ?        S<   Jan21   0:00 [kacpid]
root        29  0.0  0.0      0     0 ?        S<   Jan21   0:00 [kblockd/0]
root        50  0.0  0.0      0     0 ?        S<   Jan21   0:00 [aio/0]
root        30  0.0  0.0      0     0 ?        S    Jan21   0:00 [khubd]
root        49  0.0  0.0      0     0 ?        S    Jan21   0:00 [kswapd0]
[root@linux1 ~]# netstat -i
```

```
Kernel Interface table
Iface       MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR
Flg
eth0       1500   0   222799      0      0      0    18335      0      0      0
BMRU
eth0:1     1500   0      - no statistics available -
BMRU
lo        16436   0     3179      0      0      0     3179      0      0      0
LRU
[root@linux1 ~]# top
top - 12:50:59 up 13 days, 14:17,  2 users,  load average: 0.51, 0.12, 0.04
Tasks:  54 total,   2 running,  52 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:    449908k total,   445132k used,     4776k free,    62392k buffers
Swap:   522072k total,        0k used,   522072k free,   309496k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      16   0  3084  552  472 S  0.0  0.1   0:00.66 init
    2 root      34  19     0    0    0 S  0.0  0.0   0:14.28 ksoftirqd/0
    3 root       5 -10     0    0    0 S  0.0  0.0   0:00.00 events/0
    4 root      11 -10     0    0    0 S  0.0  0.0   0:00.01 khelper
    5 root      15 -10     0    0    0 S  0.0  0.0   0:00.00 kacpid
   29 root       5 -10     0    0    0 S  0.0  0.0   0:00.00 kblockd/0
   50 root      12 -10     0    0    0 S  0.0  0.0   0:00.00 aio/0
   30 root      15   0     0    0    0 S  0.0  0.0   0:00.06 khubd
   49 root      15   0     0    0    0 S  0.0  0.0   0:00.19 kswapd0
  196 root      22   0     0    0    0 S  0.0  0.0   0:00.00 kseriod
  318 root      15   0     0    0    0 S  0.0  0.0   0:00.22 kjournald
 1473 root      10 -10  2176  468  380 S  0.0  0.1   0:00.01 udevd
 2115 root      20   0     0    0    0 S  0.0  0.0   0:00.00 scsi_eh_0
 2116 root      15   0     0    0    0 S  0.0  0.0   1:18.07 usb-storage
 2609 root       6 -10     0    0    0 S  0.0  0.0   0:00.00 kauditd
 2650 root       6 -10     0    0    0 S  0.0  0.0   0:00.00 kmirrord
 2681 root      15   0     0    0    0 S  0.0  0.0   0:00.00 kjournald
```

Remarque : La commande `top` affiche la charge système, l'utilisation de la mémoire et les processus les plus consommateurs de ressources (%CPU, %MEM). La commande est rafraîchie périodiquement. On arrête la commande par Ctrl-C.

```
[root@linux1 ~]# top -b -d 10 -n 4 >> fic
[root@linux1 ~]# head fic
top - 13:03:10 up 13 days, 14:29,  2 users,  load average: 0.02, 0.37, 0.32
Tasks:  54 total,   1 running,  53 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.1% us,  0.0% sy,  0.0% ni, 99.8% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:    449908k total,   445260k used,     4648k free,    62464k buffers
Swap:   522072k total,        0k used,   522072k free,   309520k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      16   0  3084  552  472 S  0.0  0.1   0:00.66 init
    2 root      34  19     0    0    0 S  0.0  0.0   0:14.28 ksoftirqd/0
    3 root       5 -10     0    0    0 S  0.0  0.0   0:00.00 events/0
[root@linux1 ~]#
```

Remarque : On vient d'utiliser top en mode « batch ». On a fait 4 mesures à 10 secondes d'intervalle.

```
[root@linux1 ~]# watch free
Every 2.0s: free                                Sun Feb  4 13:04:50 2007
```

```
             total       used       free     shared    buffers     cached
Mem:        449908     445452       4456          0      62472     309584
-/+ buffers/cache:      73396     376512
Swap:       522072          0     522072
```

Remarque : On arrête la commande par Ctrl-C.

2. On utilise les commandes du package systat (sauf sar).

```
[root@linux1 ~]# rpm -q sysstat
sysstat-5.0.5-11.rhel4
[root@linux1 ~]# rpm -ql sysstat | grep bin
/usr/bin/iostat
/usr/bin/mpstat
/usr/bin/sar
[root@linux1 ~]# yum provides iostat
...
sysstat.i386                              5.0.5-11.rhel4         installed
...
[root@linux1 ~]# mpstat
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

13:53:28     CPU   %user   %nice %system %iowait    %irq   %soft   %idle
intr/s
13:53:28     all    0.07    0.00    0.04    0.04    0.02    0.00   99.83
1050.55
[root@linux1 ~]# iostat
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

avg-cpu:  %user   %nice    %sys %iowait   %idle
           0.07    0.00    0.05    0.04   99.83

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
hda               0.12         1.52         1.74    1787853    2048820
hda1              0.00         0.00         0.00       1370         30
hda2              0.26         1.49         1.71    1755778    2015048
hda3              0.00         0.00         0.00       1411         76
hda5              0.00         0.00         0.00       1074          0
hda10             0.01         0.02         0.03      20970      33650
hdc               0.00         0.01         0.00      12636          0
```

3. Installer et utiliser procinfo.

```
[root@linux1 ~]# wget 'http://downloads.sourceforge.net/mcj/procinfo-18.tar.bz2'
[root@linux1 ~]# tar xvjf procinfo-18.tar.bz2
procinfo-18/
procinfo-18/lsdev.8
procinfo-18/procinfo.8
procinfo-18/procinfo.c
procinfo-18/procinfo.h
procinfo-18/routines.c
procinfo-18/socklist.8
procinfo-18/lsdev.pl
procinfo-18/socklist.pl
procinfo-18/procinfo.lsm
procinfo-18/README
```

```
procinfo-18/CHANGES
procinfo-18/Makefile
[root@linux1 ~]# cd procinfo-18
[root@linux1 procinfo-18]# ls
CHANGES   README   lsdev.pl    procinfo.c  procinfo.lsm  socklist.8
Makefile  lsdev.8  procinfo.8  procinfo.h  routines.c    socklist.pl
[root@linux1 procinfo-18]# make
gcc -Wall -Wstrict-prototypes -O2   -c -o procinfo.o procinfo.c
procinfo.c:19: warning: 'rcsid' defined but not used
gcc -Wall -Wstrict-prototypes -O2   -c -o routines.o routines.c
routines.c:19: warning: 'rcsid' defined but not used
gcc -Wall -Wstrict-prototypes -s  procinfo.o routines.o  -ltermcap -o procinfo
[root@linux1 procinfo-18]# make install
mkdir -p /usr/bin
install procinfo /usr/bin/procinfo
install lsdev.pl /usr/bin/lsdev
install socklist.pl /usr/bin/socklist
mkdir -p /usr/man/man8
install -m 644  procinfo.8 /usr/man/man8/procinfo.8
install -m 644  lsdev.8 /usr/man/man8/lsdev.8
install -m 644  socklist.8 /usr/man/man8/socklist.8
[root@linux1 procinfo-18]# procinfo
Linux 2.6.9-42.EL (buildcentos@build-i386) (gcc 3.4.6 20060404 ) #1 1CPU
[linux1]

Memory:      Total        Used        Free      Shared     Buffers
Mem:        449908      371636       78272           0       55740
Swap:       522072           0      522072

Bootup: Sun Jan 21 22:33:28 2007    Load average: 0.00 0.00 0.00 1/54 15738

user  :       0:08:47.63   0.0%  page in :         0
nice  :       0:00:00.32   0.0%  page out:         0
system:       0:06:00.92   0.0%  swap in :         0
idle  :   2d 14:40:16.99  99.9%  swap out:         0
uptime:   2d 15:04:59.20         context :154378682

irq  0:1086255246 timer                 irq 12:        85 i8042
irq  1:       145 i8042                 irq 14:    115924 ide0
irq  4:         6                       irq 15:   9773947 ide1
irq  6:         2                       irq169:    214956 eth0
irq  7:         4                       irq185:  44604067 ehci_hcd, uhci_hcd,
irq  8:         1 rtc                   irq193:         0 VIA8233
irq  9:         0 acpi

[root@linux1 procinfo-18]# cd
```

4. Créer un script de surveillance activé par un crontab.

```
[root@linux1 ~]# vi surveille.sh
[root@linux1 ~]# cat surveille.sh
#!/bin/sh
uptime
free
ps aux |sort -rk +3 | head -10
```

```
echo "============================================================"
[root@linux1 ~]# chmod u+x surveille.sh
[root@linux1 ~]# crontab -l > /tmp/new_cron
no crontab for root
[root@linux1 ~]# echo "* * * * * /root/surveille.sh > /var/log/perf.log 2>&1" >>
/tmp/new_cron
[root@linux1 ~]# crontab /tmp/new_cron
[root@linux1 ~]# head /var/log/perf.log
 14:06:01 up 13 days, 15:32,  2 users,  load average: 0.00, 0.00, 0.00
             total       used       free     shared    buffers     cached
Mem:        449908     403816      46092          0      61948     268180
-/+ buffers/cache:      73688     376220
Swap:       522072          0     522072
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
gdm       4498  0.0  2.3 21264 10348 ?        Ss   Jan21   0:00
/usr/bin/gdmgreeter
root      4155  0.0  1.9 11944 8600 ?         S    Jan21   0:00 /usr/X11R6/bin/X
:0 -audit 0 -auth /var/gdm/:0.Xauth -nolisten tcp vt7
root      3428  0.0  1.7 11364 7952 ?         Ss   Jan21   7:55 hald
root      4133  0.0  0.5 12892 2432 ?         S    Jan21   0:00 /usr/bin/gdm-
binary -nodaemon
```

Tâche 2 :
Surveiller une commande avec time

1. Créer un programme consommateur de CPU (calcul de la constante mathématique e).

```
[root@linux1 ~]# vi e.py
# calcul de e
import sys
un_ch ="1"
for i in range(int(sys.argv[1])):
        un_ch = un_ch + "0"

un = int(un_ch)
e = un + un
n = 1
factorielle = 1

while 1:
        n = n + 1
        factorielle = factorielle * n
        e_old = e
        e = e + un/ factorielle
        if e == e_old : break

print "essai n : ", n, " e = ", e
```

2. Utiliser la commande time interne.

```
[root@linux1 ~]# time python e.py 10
essai n :  14  e =  27182818277

real    0m0.015s
user    0m0.011s
sys     0m0.005s
```

Remarques :

1) Le script de calcul de e demande en argument le nombre de décimal voulu.

2) Le temps réel est le temps pour l'être humain. Le temps CPU consommé est user+sys.

3. Utiliser la commande time externe (GNU).

```
[root@linux1 ~]# /usr/bin/time python e.py 100
essai n :  70  e =
271828182845904523536028747135266249775724709369995957496696762772407663035354759457138217852516642 38
0.00user 0.00system 0:00.01elapsed 86%CPU (0avgtext+0avgdata 0maxresident)k
0inputs+0outputs (0major+601minor)pagefaults 0swaps
```

4. Idem, mais en affichant plus d'informations.

```
[root@linux1 ~]# /usr/bin/time -v python e.py 100
essai n :  70  e =
271828182845904523536028747135266249775724709369995957496696762772407663035354759457138217852516642 38
        Command being timed: "python e.py 100"
        User time (seconds): 0.01
        System time (seconds): 0.00
        Percent of CPU this job got: 80%
        Elapsed (wall clock) time (h:mm:ss or m:ss): 0:00.01
        Average shared text size (kbytes): 0
        Average unshared data size (kbytes): 0
        Average stack size (kbytes): 0
        Average total size (kbytes): 0
        Maximum resident set size (kbytes): 0
        Average resident set size (kbytes): 0
        Major (requiring I/O) page faults: 0
        Minor (reclaiming a frame) page faults: 581
        Voluntary context switches: 1
        Involuntary context switches: 3
        Swaps: 0
        File system inputs: 0
        File system outputs: 0
        Socket messages sent: 0
        Socket messages received: 0
        Signals delivered: 0
        Page size (bytes): 4096
        Exit status: 0
[root@linux1 ~]#
```

Tâche 3 : Surveiller le système avec sar

1. Utiliser en direct la commande sar.

Dans les exemples suivants, on fait trois mesures à cinq secondes d'intervalle.

```
[root@linux1 ~]# sar -u 5 3
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

14:30:03          CPU     %user     %nice   %system   %iowait     %idle
14:30:08          all      0.00      0.00      0.00      0.00    100.00
14:30:13          all      0.00      0.00      0.00      0.00    100.00
14:30:18          all      0.00      0.00      0.00      0.00    100.00
Average:          all      0.00      0.00      0.00      0.00    100.00
```

Remarques :

1) La somme `%user+%nice+%system` donne le pourcentage d'utilisation des processeurs.

2) La somme `%iowait+%idle` donne le pourcentage d'inactivité des processeurs.

3) Le rapport `%user/%system` indique si le système fait globalement des calculs ou des IO.

```
[root@linux1 ~]# sar -q 5 3
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

14:30:30      runq-sz  plist-sz   ldavg-1   ldavg-5  ldavg-15
14:30:35            0        55      0.00      0.00      0.00
14:30:40            0        55      0.00      0.00      0.00
14:30:45            0        55      0.00      0.00      0.00
Average:            0        55      0.00      0.00      0.00
```

Remarque : `plist-sz` indique le nombre total de processus. `runq-sz` Indique le nombre de processus en attente d'un processeur.

```
[root@linux1 ~]# sar -r 5 3
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

14:31:49    kbmemfree kbmemused  %memused kbbuffers  kbcached kbswpfree
kbswpused  %swpused  kbswpcad
14:31:54        46476    403432     89.67     61980    268232    522072
0      0.00         0
14:31:59        46476    403432     89.67     61980    268232    522072
0      0.00         0
14:32:04        46220    403688     89.73     61980    268232    522072
0      0.00         0
Average:        46391    403517     89.69     61980    268232    522072
0      0.00         0
[root@linux1 ~]# sar -b 5 3
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

14:32:30          tps      rtps      wtps   bread/s   bwrtn/s
14:32:35         0.00      0.00      0.00      0.00      0.00
14:32:40         0.00      0.00      0.00      0.00      0.00
14:32:45         0.00      0.00      0.00      0.00      0.00
Average:         0.00      0.00      0.00      0.00      0.00
```

Remarque : `tps` indique le nombre de transfert entre la mémoire et les disques physiques. `rtps` indique le nombre de lectures et `wtps` le nombre d'écriture. `bread/s` indique le nombre de blocs de 512 octets lus par seconde et `wrtn/s` le nombre de blocs écrits par seconde.

```
[root@linux1 ~]# sar -B 5 3
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

15:00:36     pgpgin/s pgpgout/s   fault/s  majflt/s
15:00:41         0.00      7.19      9.98      0.00
15:00:46         0.00      0.00      6.20      0.00
15:00:51         0.00      0.00      2.00      0.00
Average:         0.00      2.40      6.07      0.00
```

Remarque : `pgpgin/s` et `pgpgout/s` indiquent les transferts en Ko entre la mémoire et le disque. `fault/s` indique le nombre de défaut de page par seconde. `majflt/s` indique le nombre de défauts de page qui entraînent une lecture disque.

```
[root@linux1 ~]# sar -R 5 3
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

15:08:07      frmpg/s   bufpg/s   campg/s
15:08:12         0.00      0.00      0.00
15:08:17         0.00      0.00      0.00
15:08:22         0.00      0.00      0.00
Average:         0.00      0.00      0.00
```

Remarque : `frmpg/s` indique le nombre de pages libérées par seconde. Une valeur négative que le système alloue des pages. `bugfpg/s` indique l'augmentation d'allocation de pages en tant que tampon disque (buffer). Une valeur négative montre une diminution de ces tampons disques.

```
[root@linux1 ~]# sar -W 5 3
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

15:08:49     pswpin/s pswpout/s
15:08:54         0.00      0.00
15:08:59         0.00      0.00
15:09:04         0.00      0.00
Average:         0.00      0.00
```

Remarque : Nombre de page-in par seconde (`pswin/s`) et de page-out par seconde (`pswpout/s`).

```
[root@linux1 ~]# ps -e |grep cupsd
16810 ?        00:00:00 cupsd
[root@linux1 ~]# sar -x 16810 5 3
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

15:19:54          PID  minflt/s  majflt/s     %user   %system   nswap/s   CPU
15:19:59        16810      0.00      0.00      0.00      0.00      0.00     0
15:20:04        16810      0.00      0.00      0.00      0.00      0.00     0
15:20:09        16810      0.00      0.00      0.00      0.00      0.00     0
Average:        16810      0.00      0.00      0.00      0.00      0.00
[root@linux1 ~]# sar -v 5 3
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

15:20:44    dentunusd   file-sz  inode-sz  super-sz %super-sz  dquot-sz %dquot-
sz  rtsig-sz %rtsig-sz
15:20:49        26708       780     26625         0      0.00         0
0.00         0      0.00
15:20:54        26708       780     26625         0      0.00         0
0.00         0      0.00
15:20:59        26708       780     26625         0      0.00         0
0.00         0      0.00
Average:        26708       780     26625         0      0.00         0
0.00         0      0.00
[root@linux1 ~]# sar -c 5 3
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

15:22:28       proc/s
15:22:33         0.00
15:22:38         0.00
15:22:43         0.00
Average:         0.00
```

Remarque : `proc/s` indique le nombre de processus créés par seconde.

```
[root@linux1 ~]# sar -n DEV 5 3
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07
...
Average:        IFACE   rxpck/s   txpck/s   rxbyt/s   txbyt/s   rxcmp/s
txcmp/s  rxmcst/s
Average:           lo      0.00      0.00      0.00      0.00      0.00
0.00      0.00
Average:         eth0      0.60      0.33     60.57     75.12      0.00
0.00      0.00
Average:         sit0      0.00      0.00      0.00      0.00      0.00
0.00      0.00
```

2. Collecter les données de sar.

```
[root@linux1 ~]# crontab -e
crontab: installing new crontab
[root@linux1 ~]# crontab -l
*/20    17-20   *       *       *       /usr/lib/sa/sa1 300 4 &
5       23      *       *       *       /usr/lib/sa2  -A  &
[root@linux1 ~]#
```

Remarque : En argument de `sa1` on indique un intervalle et un nombre de mesures. Dans l'exemple `sa1` est activé toute les 20 minutes. A chaque fois il fait 4 mesures à 300 secondes (5 minutes) d'intervalle, ce qui donne un total de 20 minutes.

3. Visualiser les données récoltées (il faut attendre au moins 5 minutes).

```
[root@linux1 ~]# sar
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

00:00:01          CPU     %user     %nice   %system   %iowait     %idle
00:10:01          all      0.03      0.00      0.03      0.00     99.94
00:20:01          all      0.03      0.00      0.03      0.00     99.94
```

Remarque : Par défaut, `sar` utilise l'option `-u` et visualise les données récoltés dans la journée.

```
[root@linux1 ~]# ls /var/log/sa/
sa04
[root@linux1 ~]# sar -u -f /var/log/sa/sa04
Linux 2.6.9-42.EL (linux1.pinguins)     02/04/07

00:00:01          CPU     %user     %nice   %system   %iowait     %idle
00:10:01          all      0.03      0.00      0.03      0.00     99.94
00:20:01          all      0.03      0.00      0.03      0.00     99.94
```

Remarque : Avec l'option `-f`, on peut voir les données récoltés un jour quelconque. Dans l'exemple, les données du 4ième jour du mois.

Tâche 4 : La comptabilité

1. Est-ce que le paquetage de comptabilité est installé ?

```
[root@linux1 ~]# rpm -q psacct
psacct-6.3.2-38.rhel4
```

2. Quel est le fichier comptable ?

```
[root@linux1 ~]# more /etc/init.d/psacct
...
# The location of the accounting file
ACCTFILE=/var/account/pacct
```

```
start() {
        [ ! -r $ACCTFILE ] && touch $ACCTFILE && chmod 600 $ACCTFILE
        action $"Starting process accounting: " /sbin/accton $ACCTFILE
        touch /var/lock/subsys/psacct
}

stop() {
        action $"Shutting down process accounting: " /sbin/accton
        rm -f /var/lock/subsys/psacct
}
...
```

3. Démarrer le service.

```
[root@linux1 ~]# /etc/init.d/psacct start
Activation de la gestion des processus :                   [  OK  ]
[root@linux1 ~]#
```

4. Dans une autre session, se connecter en tant que guest pour générer des enregistrements comptables.

```
login as: guest
Sent username "guest"
guest@192.168.218.99's password: wwii1945
Last login: Sun Feb  4 08:57:53 2007 from raichu.pokemon
[guest@linux1 ~]$ date
dim fÃ©v  4 09:06:48 CET 2007
[guest@linux1 ~]$ who
root     tty1         Jan 22 20:30
root     pts/1        Jan 30 15:20 (raichu.pokemon)
guest    pts/2        Feb  4 09:06 (raichu.pokemon)
[guest@linux1 ~]$ uptime
 09:06:54 up 13 days, 10:33,  3 users,  load average: 0.00, 0.00, 0.00
[guest@linux1 ~]$ exit
```

5. Lister les totaux de connexions.

```
[root@linux1 ~]# ac
        total        0.01
```

6. Lister les totaux de connexions par jour.

```
[root@linux1 ~]# ac -d
Today   total        0.01
```

7. Lister les totaux de connexions par utilisateur.

```
[root@linux1 ~]# ac -p
        guest                                   0.01
        total        0.01
```

8. Lister les totaux d'utilisation du CPU par commande. C'est l'option par défaut de la commande sa.

```
[root@linux1 ~]# sa
     42       2.43re       0.00cp     1054k
     12       1.53re       0.00cp     1005k   ***other
     12       0.00re       0.00cp     1186k   bash*
      6       0.00re       0.00cp     1001k   id
      4       0.00re       0.00cp      710k   ac
      2       0.90re       0.00cp     1882k   sshd*
      2       0.00re       0.00cp     1296k   initlog
```

```
      2        0.00re        0.00cp       796k   grep
      2        0.00re        0.00cp       592k   egrep
```

9. Afficher les totaux d'utilisation du CPU par utilisateur.

```
[root@linux1 ~]# sa -m
                                      43       2.43re       0.00cp     1041k
guest                                 31       1.25re       0.00cp     1039k
root                                  11       0.91re       0.00cp      986k
sshd                                   1       0.28re       0.00cp     1723k
```

10. Afficher l'utilisation du CPU pour toutes les commandes enregistrées, l'utilisateur est mentionné.

```
[root@linux1 ~]# sa -u
root       0.00 cpu      331k mem accton
root       0.00 cpu     1334k mem initlog
root       0.00 cpu     1259k mem initlog
root       0.00 cpu     1263k mem touch
root       0.00 cpu     1195k mem psacct
root       0.00 cpu      700k mem ac
sshd       0.00 cpu     1723k mem sshd             *
guest      0.00 cpu      532k mem id
...
```

11. Afficher les données brutes.

```
[root@linux1 ~]# lastcomm
sa               S     root     pts/1      0.00 secs Sun Feb  4 09:10
crond            SF    root     __         0.00 secs Sun Feb  4 09:10
sadc             S     root     __         0.00 secs Sun Feb  4 09:10
date             S     root     __         0.00 secs Sun Feb  4 09:10
sa               S     root     pts/1      0.00 secs Sun Feb  4 09:09
sa               S     root     pts/1      0.00 secs Sun Feb  4 09:09
```

12. Idem, mais concernant uniquement un utilisateur.

```
[root@linux1 ~]# lastcomm guest
sshd             SF    guest    __         0.00 secs Sun Feb  4 09:06
bash                   guest    pts/2      0.01 secs Sun Feb  4 09:06
clear                  guest    pts/2      0.00 secs Sun Feb  4 09:07
uptime                 guest    pts/2      0.00 secs Sun Feb  4 09:06
who                    guest    pts/2      0.00 secs Sun Feb  4 09:06
date                   guest    pts/2      0.00 secs Sun Feb  4 09:06
```

13. Arrêter le service.

```
[root@linux1 ~]# /etc/init.d/psacct stop
ArrÃªt de la gestion des processus :                        [  OK  ]
[root@linux1 ~]#
```

Tâche 5 : Surveiller le système en pénurie de ressources

1. Créer des commandes pour consommer des ressources.

```
[root@linux1 ~]# vi charge_cpu.sh
{ perl -e '$var=2.5;while(1){$var*=3;$var/=3;}' & sleep 10;kill $!; } &
[root@linux1 ~]# vi charge_mem.sh
{ perl -e 'for($i=0;$i<100000000;$i++){$tb[$i]=$i;}' & sleep 10;kill $!; } &
[root@linux1 ~]# vi charge_io.sh
{ find / -type f -print -exec grep -i Linux {} \; >/dev/null 2>&1 & sleep
10;kill $!; } &
```

ATTENTION : NE PAS UTILISER CES COMMANDES SUR UN SYSTEME EN EXPLOITATION !

2. Activer les commandes de simulation de charge et de surveillance en même temps.

```
[root@linux1 ~]# { sleep 10; sh charge_mem.sh & sh charge_cpu.sh & sh
charge_io.sh & } &
[1] 30019
[root@linux1 ~]# vmstat 10 5
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 0  0      0 150292  55744 182292    0    0     1     1    0    34  0  0 100  0
 4  0      0 116692  55744 182340    0    0     2     4 1052   248 19  3 78  0
 0  0      0 196236  55632 140596    0    0   131    37 1057   448 69  9 21  1
 0  0      0 196364  55632 140596    0    0     0     3 1051   143  0  0 100  0
 0  0      0 196428  55632 140596    0    0     0    17 1053   143  0  0 100  0
[1]+  Done                    { sleep 10; sh charge_mem.sh & sh charge_cpu.sh &
sh charge_io.sh & }
```

Remarque : On peut utiliser d'autres commandes de surveillance au lieu de `vmstat`.

Tâche 6 : Limiter l'usage des ressources

1. Limiter les ressources avec ulimit.

```
[root@linux1 ~]# ssh -l guest localhost
guest@localhost's password: wwii1945
Last login: Sun Feb  4 09:06:46 2007 from raichu.pokemon
[guest@linux1 ~]$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
pending signals                 (-i) 1024
max locked memory       (kbytes, -l) 32
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
stack size              (kbytes, -s) 10240
cpu time               (seconds, -t) unlimited
max user processes              (-u) 7151
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
[guest@linux1 ~]$ ulimit -S -m 32
```

Remarque : On limite la limite soft de la taille mémoire à 32 Mo.

```
[guest@linux1 ~]$ ulimit -u -m
max user processes              (-u) 7151
max memory size         (kbytes, -m) 32
[guest@linux1 ~]$ ulimit -m
32
[guest@linux1 ~]$ perl -e 'for($i=0;$i<1000000000;$i++){$tb[$i]=$i;}'
Out of memory!
[guest@linux1 ~]$ free
             total       used       free     shared    buffers     cached
Mem:        449908      25020     424888          0        252       7240
```

```
-/+ buffers/cache:       17528     432380
Swap:       522072       24180     497892
[guest@linux1 ~]$ ulimit -S -u 5
```

Remarque : On limite à 5, le nombre de processus simultané pour guest.

```
[guest@linux1 ~]$ ulimit -u -m
max user processes              (-u) 5
max memory size         (kbytes, -m) 32
[guest@linux1 ~]$ ulimit -u
5
[guest@linux1 ~]$ sleep 10 & sleep 10 & sleep 10 &
[1] 32055
[2] 32056
[3] 32057
[guest@linux1 ~]$ sleep 10 &
-bash: fork: Ressource temporairement non disponible
```

Remarque : l'activation d'un cinquième processus (ne pas oublier le shell), échoue.

```
[guest@linux1 ~]$ ulimit -u unlimited
-bash: ulimit: max user processes: cannot modify limit: OpÃ©ration non permise
[guest@linux1 ~]$ ulimit -u 751
[guest@linux1 ~]$ ulimit -m unlimited
[guest@linux1 ~]$ ulimit -u -m
max user processes              (-u) 751
max memory size         (kbytes, -m) unlimited
[guest@linux1 ~]$ ulimit -m 32
[guest@linux1 ~]$ export LANG=C
[guest@linux1 ~]$ ulimit -m unlimited
-bash: ulimit: max memory size: cannot modify limit: Operation not permitted
[guest@linux1 ~]$ ulimit -m 64
-bash: ulimit: max memory size: cannot modify limit: Operation not permitted
[guest@linux1 ~]$ exit
```

Remarque : Quand on modifie la limite hard, on ne pas revenir en arrière (pour cette session).

2. Limiter les ressources de guest à chaque connexion.

```
[root@linux1 ~]# cp /etc/security/limits.conf /etc/security/limits.conf.000
[root@linux1 ~]# vi /etc/security/limits.conf
...
#@student        -       maxlogins       4
guest            -       maxlogins       1
guest            hard    nproc           5
# End of file
```

Remarque : On ajoute les lignes en italique.

```
[root@linux1 ~]# ssh -l guest localhost
guest@localhost's password: wwii1945
Last login: Sun Feb  4 17:02:10 2007 from localhost.localdomain
[guest@linux1 ~]$ sleep 30 & sleep 30 & sleep 30 &
[1] 32104
[2] 32105
[3] 32106
[guest@linux1 ~]$ sleep 10 &
-bash: fork: Ressource temporairement non disponible
[guest@linux1 ~]$
```

A partir, d'un autre terminal, on essaye de se connecter, cela échoue.

```
[root@linux1 ~]# ssh -l guest localhost
guest@localhost's password: wwii1945
Read from remote host localhost: Connection reset by peer
Connection to localhost closed.
```

On remet la configuration d'origine.

```
[root@linux1 ~]# cp /etc/security/limits.conf.000 /etc/security/limits.conf
cp: overwrite `/etc/security/limits.conf'? y
[root@linux1 ~]#
```

3. Supprimer la mise à jour des dates de derniers accès.

```
[root@linux1 ~]# dd if=/dev/zero of=/root/GROS_FIC bs=1k count=100000
100000+0 records in
100000+0 records out
[root@linux1 ~]# mkfs -q -F /root/GROS_FIC
[root@linux1 ~]# mount -o loop /root/GROS_FIC /mnt
[root@linux1 ~]# cal > /mnt/f1
```

On attend un peu (deux minutes environ).

```
[root@linux1 ~]# cat /mnt/f1 > /dev/null
[root@linux1 ~]# ls -l /mnt/f1
-rw-r--r--  1 root root 137 Feb  4 17:40 /mnt/f1
[root@linux1 ~]# ls -lu /mnt/f1
-rw-r--r--  1 root root 137 Feb  4 17:41 /mnt/f1
```

Remarque : La lecture du fichier à provoqué la mise à jour de la date de dernier accès et donc à provoqué UNE ECRITURE SUR LE DISQUE !

On recommence l'exercice mais en désactivant la mise à jour des dates de dernier accès.

```
[root@linux1 ~]# umount /mnt
[root@linux1 ~]# mount -o loop,noatime /root/GROS_FIC /mnt
[root@linux1 ~]# date
Sun Feb  4 17:46:52 CET 2007
[root@linux1 ~]# cat /mnt/f1 > /dev/null
[root@linux1 ~]# ls -lu /mnt/f1
-rw-r--r--  1 root root 137 Feb  4 17:41 /mnt/f1
[root@linux1 ~]# umount /mnt
[root@linux1 ~]#
```

Tâche 7 : Modifier les paramètres du noyau.

1. Créer des scripts

a) Un script qui simule une application qui écrit sur disque.

```
[root@linux1 ~]# vi load_output.sh
#!/bin/sh
GROSFIC=/usr/grosfic.dat
for i in 1 2 3 4 5 6 7
do
        dd if=/dev/zero of=$GROSFIC bs=1k count=100000
        rm $GROSFIC
done
```

b) Un script de surveillance de la mémoire.

```
[root@linux1 ~]# vi visu_charge_memoire.sh
```

```
#!/bin/sh
for i in 1 2 3 4 5 6 7
do
        clear
        date
        head -20 /proc/meminfo
        sleep 3
done
```

2. Activer les écritures disque et surveiller l'activité mémoire.

a) Avant de changer un paramètre du noyau.

```
[root@linux1 ~]# cat /proc/sys/vm/page-cluster
3
[root@linux1 ~]# time sh load_output.sh > /tmp/result 2>&1 &
[1] 32416
[root@linux1 ~]# sh visu_charge_memoire.sh
Sun Feb  4 18:17:36 CET 2007
MemTotal:       449908 kB
MemFree:        279088 kB
Buffers:          4076 kB
Cached:         130568 kB
SwapCached:       6032 kB
...
```

b) Après (on prend l'exemple du paramètre page-cluster).

```
[root@linux1 ~]# echo 20 > /proc/sys/vm/page-cluster
[root@linux1 ~]# cat /proc/sys/vm/page-cluster
20
```

On refait l'exercice précédent.

Tâche 8 : Simuler une charge avec un benchmark, l'influence de la mémoire

On simule une charge Web avec l'outil ab (Apache doit être actif).

```
[root@linux1 ~]# /etc/init.d/httpd restart
ArrÃªt de httpd :                                           [ÃCHOUÃ]
DÃ©marrage de httpd :                                       [  OK  ]
[root@linux1 ~]# ab -c 20 -n 10000 'http://localhost/'
This is ApacheBench, Version 2.0.41-dev <$Revision: 1.141 $> apache-2.0
Copyright (c) 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Copyright (c) 1998-2002 The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Finished 10000 requests
```

```
Server Software:        Apache/2.0.52
Server Hostname:        localhost
Server Port:            80

Document Path:          /
Document Length:        5044 bytes

Concurrency Level:      20
Time taken for tests:   6.656780 seconds
Complete requests:      10000
Failed requests:        0
Write errors:           0
Non-2xx responses:      10010
Total transferred:      52472420 bytes
HTML transferred:       50490440 bytes
Requests per second:    1502.23 [#/sec] (mean)
Time per request:       13.314 [ms] (mean)
Time per request:       0.666 [ms] (mean, across all concurrent requests)
Transfer rate:          7697.72 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    4   2.1      5      23
Processing:     4    7   2.6      7      22
Waiting:        1    5   2.2      6      21
Total:          9   12   1.4     12      30

Percentage of the requests served within a certain time (ms)
  50%     12
  66%     13
  75%     13
  80%     13
  90%     13
  95%     13
  98%     14
  99%     18
 100%     30 (longest request)
[root@linux1 ~]#
```

Remarques :

1) Dans l'exemple, la commande `ab` active 10000 requêtes HTTP, on active 20 requêtes simultatnément (cf. ab(8).

2) On peut activer la commande ab à partir du poste en binôme.

On réalise le test :

a) En mode graphique avec 128 Mo de mémoire.

b) En mode texte avec 129 Mo de mémoire.

c) En mode graphique dans l'état par défaut du serveur.

Remarque : Pour indiquer le niveau d'init et la mémoire, on édite la configuration Grub lors du démarrage, par exemple :

kernel /vmlinuz-2.6.9-11.EL.smp ro root=LABEL=/ rhgb quiet ***3 mem=128M***